

End-to-End (E2E) Testing

System testing

Acceptance testing

Acceptance Testing

- **Testing the entire system as a whole.**
 - **UI + Server-side + Database**
- **Concerns:**
 - **Functionality. ******
 - **From the USER interface perspective.**
 - **Performance.**
 - **Load/Stress.**

E2E Testing

- **Many similarities to API testing:**
 - **Blackbox – not looking at internals, only expected output for specific inputs.**
 - **May also be interested in side-effects, e.g. database changes.**
 - **The Asynchronuous nature (for web/mobile apps).**
- **Unit and Integration test should have ironed out (most) 'low level' errors.**

E2E Testing

- **Web apps - Targeting the browser interface.**
 - **Form submits.**
 - **Navigation.**
 - **Flows e.g. shopping cart checkout.**

Automation Tools

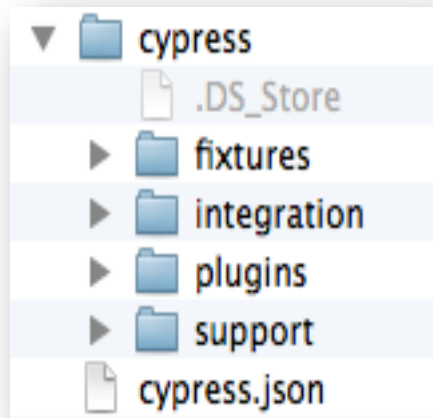
- **Traditional tool suite: Mocha + Chai + Selenium.**
- **(Very) modern tool suite: Cypress**
 - **Uses Mocha and Chai internally.**
- **Cypress.**
 - **Win / Mac / Linux**
 - **MIT License**
 - **Open Source**

Cypress

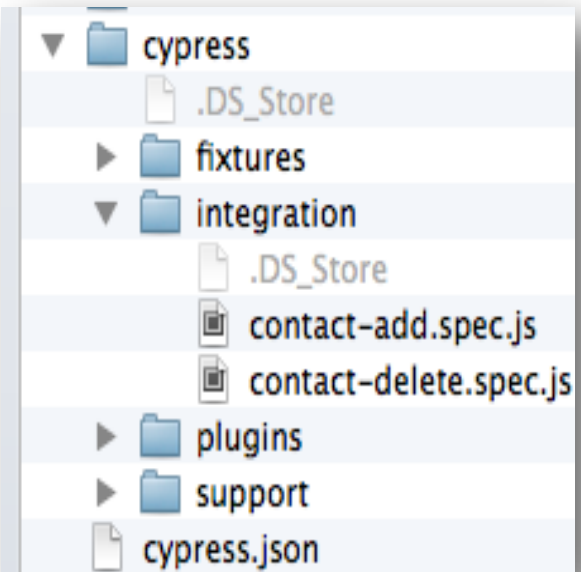
- **Getting started:**

`$ npm install --save-dev cypress`

- **(Default) Test code folder structure:**



- **fixtures** – mock data
- **Integration** – test code, termed specs.
- **cypress.json** – config file



- **CLI (Command Line Interface) has 2 main commands:**

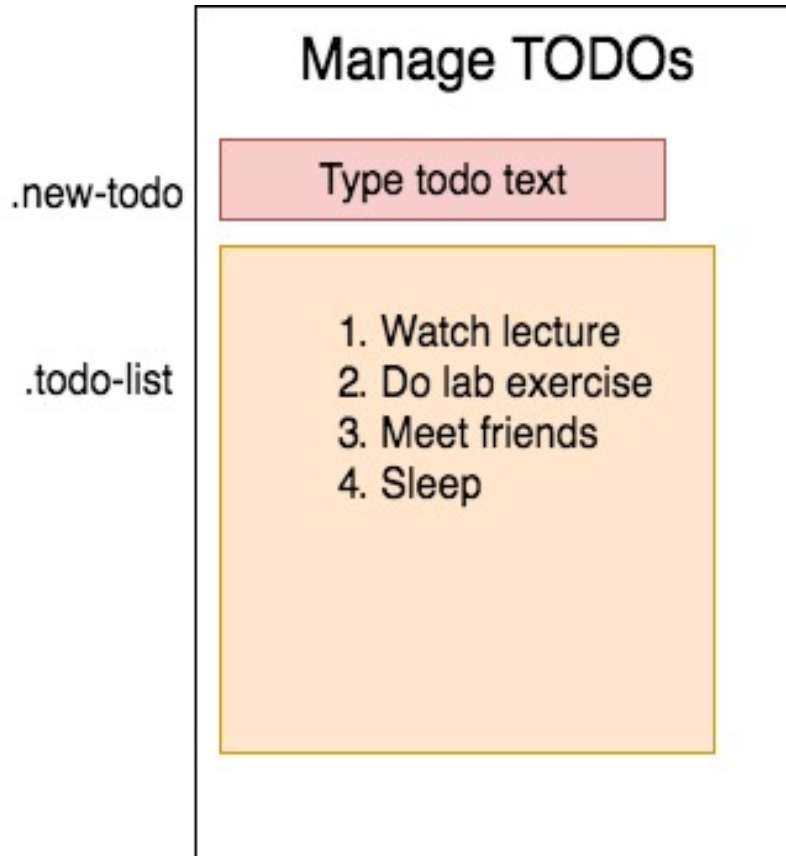
`$ npx cypress open`

- **GUI interactive mode (Dev)**

`$ npx cypress run`

- **headless mode (Testing/CI).**

Sample test code



```
describe("TODO app", () => {  
  it('adds 2 todos', () => {  
    cy.visit('http://localhost:3000')  
    cy.get('.new-todo')  
      .type('learn testing{enter}')  
      .type('complete lab{enter}')  
    cy.get('.todo-list li')  
      .should('have.length', 2)  
  })  
})
```

- **Method Chaining style e.g.**
`cy(...).get(...).type(...)`

Cypress statements

`cy.get(...selector ...)` `.should(...assertion/expectation)`

Command

Assertion (Expectations)

- **Commands:**

- get()** - **Get one or more DOM elements by selector**

- contains(text)** - **Get the DOM element containing the text,**
e.g. `cy.contains('Welcome')`

- find()** - **Get the descendent DOM elements of a selector.**

- e.g. `cy.get('.article').find('button')` // Yield the 'button' within '.article'

- click()** – **click a DOM element,** e.g. `cy.get('button').click()`

- select()** - **Select an <option> within a <select>**

- e.g. `cy.get('#paymenttype').select('Visa')`

- See <https://docs.cypress.io/api/api/table-of-contents.html>

Cypress statements

cy.get(. . .selector . . .).should(...assertion/expectation)

- **Selector: Based on CSS/JQuery style.**
 - Id**, e.g. `cy.get('#heading')`
 - CSS Class**, e.g. `cy.get('.info-message')`
 - Tag**, e.g. `cy.get('input')`
 - Attributes**, `cy.get('button[type=submit]').click()`
 - The data-test attribute.**
 - nth-child**, e.g. **get the 8th column of the 3rd row in a table**
`cy.get('tbody').find('tr:nth-child(3)').find('td:nth-child(8)')`
 - These can be combined**, e.g. `div.container` (the `div` tag with CSS class `.container`)

Cypress Test Runner

- **Main features:**
 - **Tests run** inside the browser.
 - **Full access to browser resources** – DOM, cookies, local storage.
 - **Framework-agnostic.**
 - **Flake-free test execution.**
 - **Auto retries commands** (e.g. `get()`) to cope with slow DOM construction.
 - **Deals with unpredictable nature of the web.**
 - **Supports time-travel for convenient debugging.**

\$ npx cypress open

(Interactive runner)

The screenshot displays the Cypress interactive runner interface. The main window shows a web application running on `localhost:8080` with the URL `http://localhost:8080/_/#/tests/integration/contact-add.spec.js`. The application features a contact management interface with a table of contacts and a form to add new ones.

On the left, the Cypress command log shows the test suite `contact-add.spec.js` under the heading `INTEGRATION TESTS`. A red arrow points to the `allows a contact be added` test, which is currently passing. The test steps are as follows:

- 1 GET span.badge
- 2 -CONTAINS 5
- 3 GET input[placeholder=Name]
- 4 -TYPE user X
- 5 GET input[placeholder=Address]
- 6 -TYPE 22 main street
- 7 GET input[placeholder="Phone No."]
- 8 -TYPE 055 123456
- 9 GET button
- 10 -CONTAINS Add Contact
- 11 -CLICK
- 12 GET span.badge
- 13 -CONTAINS 6

The web application interface shows a table with 5 contacts:

Contact 1	Contact 2	Contact 3	Contact 4	Contact 5
123 Test St 132-3212	23 Main St 934-4329	4 Lower St 432-5832	49 Upper Street 934-4290	4 High Street 933-3390

Each contact card includes an 'Edit' button and a 'Delete' button. The bottom of the interface shows the Cypress version `Version 3.1.5 | Changelog`.

- **Time-travel – Step through test code to track UI state.**

The screenshot displays the Cypress test runner interface. On the left, the 'Tests' panel shows a test suite 'Contact Add' with a single test 'allows a contact be added'. The test steps are listed, and the step '- TYPE 22 main street' is highlighted with an orange box. An orange arrow points from this step to the corresponding input field in the application UI. The application UI, titled 'contactList', shows a form with 'Add Contact' and 'user X' buttons, and a list of five contacts. Contact 2's address field is highlighted with an orange box, and an orange arrow points from the highlighted step in the test runner to this field. The bottom of the interface shows a 'DOM Snapshot: after' label.

Chrome File Edit View History Bookmarks People Window Help

localhost:8080/_/#/tests/integration/contact-add.spec.js

Tests 1 2.62 http://localhost:8080/ 1000 x 660 (93%)

Contact Add

allows a contact be added

BEFORE EACH

1 VISIT /

TEST

1 GET span.badge

2 - CONTAINS 5

3 GET input[placeholder=Name]

4 - TYPE user X

5 GET input[placeholder=Address]

6 - TYPE 22 main street

7 GET input[placeholder=Phone No.]

8 - TYPE 055 123 456

9 GET button

10 - CONTAINS Add Contact

11 - CLICK

12 GET span.badge

13 - CONTAINS 6

Add Contact user X

Phone No.

Contact 1

Contact 1

123 Test St

132-3212

Edit Delete

Contact 2

Contact 2

23 Main St

934-4329

Edit Delete

Contact 3

Contact 3

4 Lower St

432-5832

Edit Delete

Contact 4

Contact 4

49 Upper Street

934-4290

Edit Delete

Contact 5

Contact 5

4 High Street

933-3390

Edit Delete

DOM Snapshot: after

Selectors

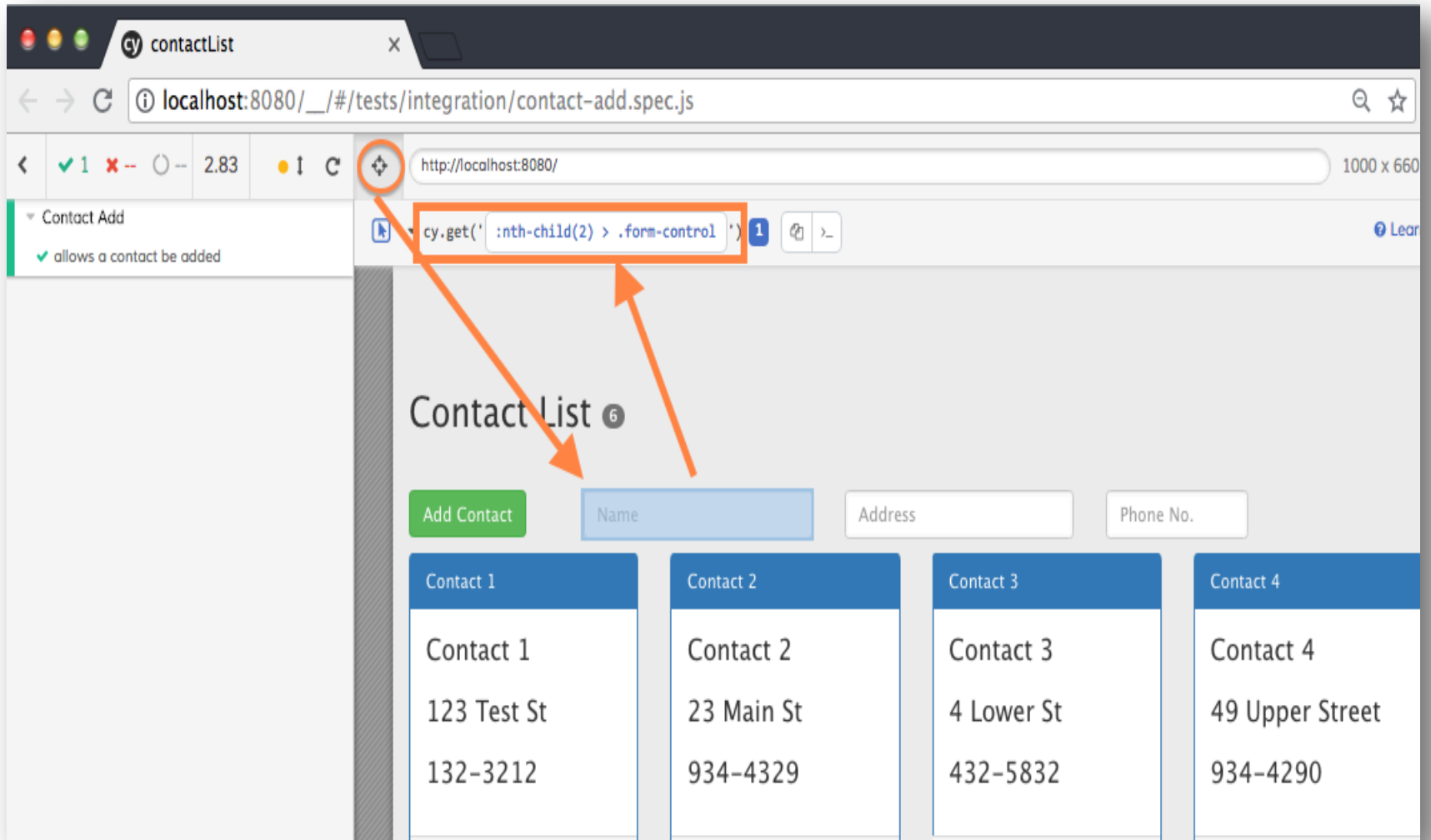
cy.get(...selector...).should(...assertion/expectation)

- **Has impact on test brittleness.**
 - **Small changes to CSS or HTML structure can cause test failure.**
- **Use data-test attribute, where possible, to avoid brittle tests, e.g.**

```
<input data-test="name" type="text" className="form-control" onChange={....} />
```

```
cy.get('input[data-test=name]').type("Joe Bloggs")
```

- Selector Playground – **good learning aid.**



Cypress Test Runner

- **Headless mode:**
\$ npx cypress run
- **Runs all tests.**
- **Ideal for CI (Continuous Integration) environment.**
- **Generates video recordings (default; configurable).**
 - **.mp4 file type.**
 - **Facilitates sharing and project visibility.**
 - **Dashboard service (Publish recordings)**

Commands are in a queue

```
it('adds 2 todos', () => {  
  cy.visit('http://localhost:3000')  
  cy.get('.new-todo')  
    .type('learn testing{enter}')  
    .type('complete lab{enter}')  
  cy.get('.todo-list li')  
    .should('have.length', 2)  
})
```

